In: *The Story of Managing Projects: A Global, Cross- Disciplinary Collection of Perspectives*,
Dr. E. G. Carayannis and Dr. Y. H. Kwak, editors,
Greenwood Press / Quorum Books, 2002

Chapter X.

AGILE PROJECT MANAGEMENT METHODS FOR IT PROJECTS

Glen B. Alleman
Niwot, Colorado 80503

> The difference between failure and success is the difference between doing something almost right and doing something right.
>
> — Benjamin Franklin

Agile project management methodologies used to develop, deploy, or acquire information technology systems have begun to enter the vocabulary of modern organizations. Much in the same way *lightweight* and *agile* manufacturing or business management processes have over the past few years. This chapter is about applying *Agile* methods in an environment that may be more familiar with high ceremony project management methods – methods that might be considered heavy weight in terms of today's *agile* vocabulary.

High ceremony projects include projects based on formal or semi- formal project management methods, ones like Prince2 [1], PMI's PMBOK [2], or processes based on the Software Engineering Institute's Capability Maturity Model [3]. These methods are traditionally associated with organizations that operate in *software engineering* centric business domains. These domains view *software* activities as an engineering process, rather than a creative process based in the skill of individuals or small teams.

Organizations with mature processes often define their activities in a formal manner, applying methods with rigor, and monitoring the processes and results carefully. These practices are many times built up over time and come about through direct experiences – either good or bad. Many times, they follow the formal structure of the underlying business process.

---

[1] *P*rojects *IN* *C*ontrolled *E*nvironments (PRINCE) is a structured project management method used in the United Kingdom.

[2] Project Management Body of Knowledge (PMBOK) is an ANSI Standard PMI- 99- 001- 2000 describing the various attributes of a project management method.

[3] Capability Maturity Model is a collection of *model frameworks* for assessing the maturity of a specific practice. Key Practice Areas are used to define the various levels of maturity. CMM now consists of: Software, People, Software Acquisition, Systems Engineering, and Integrated Product Development. This models are supported by a software process assessment standard ISO/IEC 15504.

It is common to talk about *Agile* methods for modern project management processes in the context of a set of *lightweight* activities used to manage the development or acquisition of software. These activities include requirements, design, coding, and testing processes based on a minimal set of activities needed to reach the end goal — a working software system. [4]

Although some of these agile development methods address the *management* aspects of software projects – people, processes, and technology – they are primarily focused on coding, testing, and software artifact delivery. [5]

Applying the concept of *agility* to the management of a software project is a natural step in the evolution of software development. One important question to be asked though is *how can these minimalist approaches be applied to traditional project management activities*?

---

[4] SCRUM, DSDM, Crystal, Adaptive Software Development, and Extreme Programming.

[5] Some proponents of these lightweight methods contend delivery of software is the only goal. Although this is an obvious outcome of the programming and integration process, there are many other deliverable artifacts associated with large- scale software projects.

- What project management process simplifications are *appropriate* for a specific problem domain? [6]
- Are all lightweight and agile project management process steps applicable to specific problem domains? If not which steps are applicable to which domains? [6]

Weight versus Agility

In the information technology project management literature, *lightweight* is often defined as *not heavyweight*, which is a tautology. Over time *lightweight* has been superceded in the trade press and literature by the term *Agile*. *Lightweight* and *Agile* are not interchangeable words however. This distinction is not well understood by many *agile* proponents, so some clarification is needed here before we proceed.

Lightweight describes the weightiness of the process and its artifacts. The amount of potentially *non-value added* artifacts produced by a specific process. This weightiness can be attributed to the undesirable conse-

[6] Capers Jones' *Software Assessments, Benchmarks, and Best Practices* provides a taxonomy that is useful for defining the various domains. Management Information Systems, Outsourced systems, Systems software, Commercial software, Military software, End User software, and web applications and e-project.

quences of the process – artifacts that don't provide benefit to the outcome. This weightiness can also be attributed to the mis- application of a specific process. *Agility* describes the behavior of the participants and their ability to *move* or *adjust* in new and possibly unforeseen situations.

Much like an overweight boat, airplane or athlete, the undesirable weight needs to be removed in order to increase the efficiency of the vehicle. This is a standard *best practice* in many engineering disciplines. One problem with this analogy though is that anyone suggesting a specific methodology is *over weight* must answer the question:

> … if a project management method were properly applied, in the proper domain, to the proper set of problems, with properly trained participants, would it be considered overweight and produce undesirable consequences?

The usual answer is *no, of course not*. If everyone were doing their job properly, in the proper engineering, regulatory, and contractual environment, then the results would be accepted by all the participants – this is the definition of a tautology.

The problem of *Agile* project management methodology selection is compounded by the behaviors of the method as well as the behaviors of the participants using the method. In addition, the *appropriateness* of the method for a specific problem domain remains an issue. Making a process *lightweight* by removing activities or artifacts is most likely inappropriate and a possible source for project failure without careful consideration of the consequences.

Project Management Framework

According to the Software Engineering Institute, a methodology must posses certain attributes in order to meet the requirements of being called a methodology. [7] Another framework for methodologies is the Software Engineering Body of Knowledge (SWEBOK) that contains other knowledge development methods to be used by any professional software engineer. [8] For the moment, we'll focus on the SEI's description of the software project attributes. Figure 1 describes how these at-

[7] "Software Development Taxonomy," www.sei.cmu.edu/legacy/kit/taxonomy.html.

[8] www.swebok.org

tributes could be related in an *agile* project management method. [9] This structure is a process pattern view of project activities. [10] This approach focuses on the communication and people-centric aspects of project management. *Agile* project management can be built on this framework.

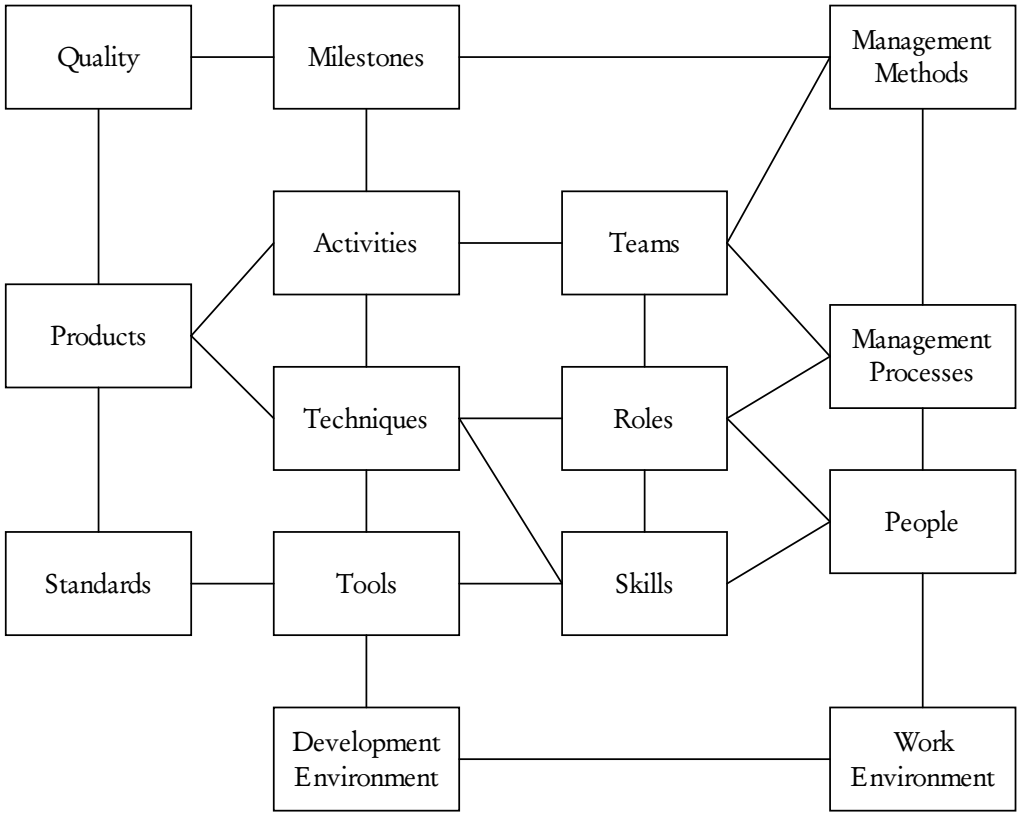

Figure 1 – Interrelation between Project Management Activities

[9] "A Methodology per Project," Alistair Cockburn, Humans and Technology Technical Report, TR 99.04, October 1999. http://www.crystalmethodologies.org/articles/mpp/methodologyperproject.html

[10] One source of process patterns is http://i44pc48.info.uni-karlsruhe.de/cgi-bin/OrgPatterns.

The *Agile* Software Delivery Process

Agile processes emphasize both the rapid and flexible adaptation to changes in the process, the product, and the development environment [11]. This is a very general definition and therefore not very useful without some specific context.

Before establishing this context, *Agile* processes include three major attributes, they are:

- *Incremental and Evolutionary* – allowing adaptation to both internal and external events.
- *Modular and Lean* – allowing components of the process to come and go depending on specific needs if the participants and stakeholders.
- *Time Based* – built on iterative and concurrent work cycles, which contain feedback loops and progress checkpoints.

---

[11] "New Age of Software Development: How Component Based Software Engineering Changes the Way of Software Development," Mikio Aoyama, 1998 International Workshop on Competent Based Software Engineering, 1998.

"Agile Software Process and Its Experience," Mikio Aoyama, International Conference on Software Engineering, 1998.

Common Problems with All Software Projects

The National Software Quality Experiment has been conducted each year since 1992 with the following observations reoccurring over the years: [12]

Common Problem	Consequences
Software product source code components are not traced to requirements.	Software product is not under the control and the verification procedures are imprecise. Changes cannot be managed in a controlled manner.
Software engineering practices are not applied in a systematic manner.	Defect rates are unacceptable.
Product designs and source are managed in an ad hoc manner	The understandability, maintainability, and adaptability of the product is negatively impacted.
The construction processes for the product are not clearly defined.	Common patterns of the processes are not exploited.
Rapidly changing code base has become the norm.	The code base services the only the short term benefits and mortgages the future where traceable baseline requirements, specifications, and design artifacts are the foundations of success.

Figure 2 – National Software Quality Experiment Results

[12] In 1992, the DOD Software Technology Strategy set the objective to reduce software problem rates by a factor of ten by the year 2000. The National Software Quality Experiment is a mechanism for obtaining core samples of software product quality. This national database provides the means to benchmark and measure progress towards the national software quality objective and contains data from 1992 through 1998.

The centerpiece of the experiment is the Software Inspection Lab where data collection procedures, product checklists, and participant behaviors are packaged for operational project use. The uniform application of the experiment and the collection of consistent measurements are guaranteed through rigorous training of each participant.

The Problem of Change

Change continually takes place in the business world, at all levels within an organization or market place. Change by itself is not the problem. The world is always changing. It always has been changing. It always will be changing. Businesses and the processes they use have always had to adapt to this changing world.

Often changes in the past have occurred incrementally. When a radical change took place, the next change *event* was slow in coming. While there has always been uncertainty in business, it was usually not significant or sustained.

The problem in today's world is that change is no longer incremental or even linear. Radical *non- linear changes* occur in the normal course of business. The pace of change is not only increasing, sustained uncertainty is now commonplace.

Ready For Agility?

All organizations face problems that can be addressed by *Agile* methods, but not all companies are ready for the radical ideas needed to become an *Agile* organization. Agility is still an emerging topic and is at the

stage where it is not possible to buy an off-the-shelf solution that has been shown to behave in the same manner as heavier weight processes. [13] Elements of agility can certainly be found in many processes, but as the saying goes – *one swallow does not a make summer*. [14]

The introduction of an *Agile* process should only be undertaken by organizations that are *risk* aware if not *risk adverse*. Organizations who need answers and concepts that are fully developed that result in a solution that can be implemented with little risk should stay clear of the *Agile* Processes. The irony is though – there is no such process that can deliver a fully developed plan that results in a fully developed project or product. Let alone one that can be deployed without risk.

---

[13] This of course is not the contention of XP, SCRUM, ASD, and other lightweight, and now *agile* processes. But these processes have yet to enter the stage where analytical evidence has been gathered to support the contention they produce superior results when compared to their less-lightweight cousins. This is a continuing debate which will not be resolved in this short chapter.

[14] This English proverb can be traced to a Greek proverb. In the ancient world birds were associated with the household gods and their presence was looked upon as fortuitous. Any harm done to them would bode evil for the household.

The Forces Driving Agility

Software acquisition and deployment is generally driven by a need to solve a specific problem, to do things better, to modify or improve a business or technical process. The software development process community has two *schools of thought* regarding the outcome of these efforts:

- Things are getting better
- Things are getting worse

This conflicting set of *opinions* adds more confusion to an already confusing question of — *are we actually improving the outcome of the software by improving the management processes*?

A few years ago, methodologies and processes were the domain of academics. The methodology *zoo* has grown however and at the same time become focused on the commercial aspects of selling these methodologies to anxious managers, developers, and stakeholders. This *selling process* has, in many cases, overtaken the rational application of these methods of specific problem domains.

Practical Agile Project Management

The deployment of an *Agile* project management methodology in an existing organization faces several obstacles:

- The legacy project management processes must be displaced in some way to make room for the new process.
- The gaps that existed in the legacy process must be filled with the new process while maintaining the integrity provided by the legacy process.

Common Threads of These Methods

In an attempt to simplify the many attributes of the methods, a list of common threads can be built, using Figure 1 as a framework.

Thread	Compliance
Requirements Gathering	Some method of gathering requirements is needed.
Software Development or Procurement	Software must be developed (or procured) that meets the requirements.
Testing	Component and System testing are performed in some structured manner.
Personnel Management	The management of personnel is provided in some methods, but not all.
Project Management	Some means of defining tasks, measuring progress, providing feedback, and changing the course of the participants.

Figure 3 – Common Aspects of All Methods

Is Agile Yet Another Software Methodology Taxonomy?

Before selecting a software development method, some understanding of what type of software is to be developed is appropriate. [15] By partitioning software system into types, the *appropriateness* question can be addressed — *what project management methods are appropriate for what problem domains*?

Software Type	Attributes
Management Information Systems	Software that an enterprise uses to support business and administrative operations.
Outsourced systems	Software projects built for client organizations.
Systems software	Software that controls a physical device such as a computer or a telephone switch.
Commercial software	Software applications that are marketed to hundreds or even millions of clients.
Military software	Software produced for the uniformed services.
End User Software	Small applications written for personal use.
Web Application and e-Projects	Small, medium, and large scale projects with legacy system integration, transaction processing, multimedia delivery, and web browser based user interfaces

Figure 4 – Software Systems Taxonomy

[15] *Software Assessments, Benchmarks, and Best Practices*, Capers Jones, Addison Wesley, 2000. This partitioning is not unique, but it is based on an underlying assumption that there are fundamental differences between problem domains.

Framework for Agile Project Management Methods

A framework for deploying *Agile* project management processes provides a descriptive guideline rather than a proscriptive set of rules. This framework approach provides the user a broader set of recommendations than is found in any particular named methodology.

This framework is based on two foundations:

- The Software Program Managers Network Nine Best Practices [16] – which provides guidelines for project managers using practical suggestions for daily application.
- Scott Ambler's *Agile Modeling* framework – which provides a broad framework for creating *agile* processes applied to software projects. [17]

---

[16] www.spmn.com

[17] "Agile Modeling," Scott Ambler, www.agilemodeling.com. *Process Patterns: Building Large–Scale Systems Using Object Technology*, Scott Ambler, Cambridge University Press, 1998. *More Process Patterns: Delivering Large-Scale Systems Using Object Technology*, Scott Ambler, Cambridge University Press, 1999.

Agile Project Management Guidelines

Building on the Software Program Managers *Nine Best Practices*, the well established project management methods of the past, the fundamentals of any project management method, and finally common sense, a framework for *Agile Project Management* can be built.

Values of Agile Project Management

Before the principles of *Agile Project Management* can be defined, a set of underlying *values* are useful.

- *Communication* – of information within and outside an *Agile* project is constant. It is the responsibility of the project manager to ensure that the communication occurs effectively, clearly, and in a timely manner between the management, the contributors, and the stakeholders. Since *change* is constant in an *Agile* project, constant communication is the only means of maintaining the connections between all the participants. *Going Dark* for any significant amount of time is simply not allowed.

- *Simplicity* – defines the approach to identifying the *critical success factors* of the project in terms of the simplest possible solution. All

activities must contribute a measurable value to the project management process. Measuring the value of a project management artifact is the role of the stakeholders and the project manager. This is done by asking *what is the value of this specific task, artifact, or project deliverable*?

- *Feedback* – "optimism is an occupational hazard of software development, feedback is the cure". [18] Continuous feedback is a primary tool for defining and sustaining *Agility*.
- *Courage* – all important decisions and changes in the direction of the project need to be made with the courage. Change is part of any realistic IT project. Dealing with the consequences of change or discarding the outcome when the decision is proven inadequate requires courage.
- *Humility* – the best project managers acknowledge they don't know everything. The stakeholders, project participants, and customers all have their our area of expertise and add value to the

[18] *Extreme Programming Explained*, Kent Beck, Addison Wesley, 1999.

project. An effective approach is to assume that everyone involved with the project has equal value and therefore should be treated with respect.

Applying Agile Principles in Practice

Applying these principles in practice creates the foundation for managing IT projects in an Agile manner.

- *Assume Simplicity* – as the project evolves it should be assumed that the simplest solution is the best solution. [19] Overbuilding the system or any artifact of the project must be avoided. The project manager should have the courage to not perform a task or produce an artifact that is not clearly stated in the requirements *as needed for the immediate benefit of the stakeholders*.

- *Embrace Change* – since requirements evolve over time. The stakeholder understanding of the requirements will change over time. Project stakeholders themselves may change as the project makes progress. Project stakeholders may change their point of view,

[19] This may not always be the case but it is a good starting point.

which in turn will change the goals and success criteria of the project management effort.

- *Enabling The Next Effort Is Also A Goal* – the project can still be considered a failure even when the team delivers a working system to the users. Part of fulfilling the needs of the project stakeholders is to ensure that the system is robust enough to be extended over time. Using Alistair Cockburn concept, "when you are playing the software development game your secondary goal is to setup to play the next game." [20] The next phase may be the development of a major release of the system or it may simply be the operation and support of the current version of the system.
- *Incremental Change* – the pressure to *get it right the first time* can overwhelm the best project manager. Instead of futilely trying to develop an all encompassing project plan from the start, put a stake in the ground by developing a small portion of the system, or even a high- level model of a larger portion of the system, and

[20] http://crystalmethodologies.org/

evolve this portion over time. Or simply discard it when you no longer need it in an incremental manner.

- *Maximize Stakeholder Value* – the project stakeholders are investing resources — time, money, facilities, etc. — to have a system deployed that meets their needs. Stakeholders expect that their investment to be applied in the best way.

- *Manage With A Purpose* – create artifacts of the project management process that have stakeholder value. Identify why and for whom the artifact is created. Identify a valid purpose for creating the artifact and the audience for that artifact. This principle also applies to a change to an existing artifacts.

- *Multiple Project Views* – provide different views of the same process for different audiences. Considering the complexity of any modern information technology system construction or acquisition process, there is a need for a wide range of presentation formats in order to effectively communicate with the stakeholders, participants, and providers.

- *Rapid Feedback* – the time between an action and the feedback on that action must be minimized. Work closely with the stakeholders, to understand the requirements, to analyze those requirements, and develop a *actionable* plan, which provides numerous opportunities for feedback.

- *Working Software Is The Primary Goal of the Project* – the goal of any software project is to produce software that meets the needs of the project stakeholders. The goal is not to produce extraneous documentation, management artifacts, or even models of these artifacts. Any activity that does not directly contribute to the goal of producing a working system should be examined.

- *Travel Light* – every artifact that is created, and kept, will need to be maintained over it's life cycle. The effort needed to maintain these artifacts must be balanced with their value. Not only must the effort be considered, but the risk that the artifact will create confusion over time if it is not properly maintained must be considered.

Recommendations to Practitioners

- Make incremental change to the requirements, project plan system, and the resulting artifacts to enable agility.
- Strive for rapid feedback to ensure the project meets the needs of all the participants and stakeholder.
- Manage with a purpose, performing only those tasks that add value to business processes supported by the system.
- Travel light, discarding processes and artifacts that don't add enduring value to the product — a working software system.